Application No.: 10/010389 Docket No.: SMQ-143/P6594

AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method for calculating the number of valid instructions within a microprocessor, comprising:

advancing instructions along a microprocessor pipeline; ~~and~~

edge detecting a transition from a valid ~~instructions~~ to an invalid instruction within the microprocessor pipeline[[.]]; and

determining the number of valid instructions based on the edge detection of the valid instructions.

2. (Currently Amended) A method for calculating the number of valid instructions within a microprocessor, comprising:

fetching a bundle of instructions; ~~and~~

~~edge detecting valid instructions within the bundle.~~

edge detecting a transition from a valid instruction to an invalid instruction in the bundle of instructions; and

determining the number of valid instructions up to the transition from the valid instruction to the invalid instruction.

3. (Previously presented) A method according to claim 2, further comprising shifting at least one instruction within the bundle.

4. (Previously Presented) A method according to claim 3, further comprising shifting at least one instruction based at least in part on the number of valid instructions in the bundle.

5. (Previously Presented) A method according to claim 3, further comprising compressing the bundle of instructions.

6. (Previously Presented) A method according to claim 3, further comprising compressing the bundle of instructions for a monotonic instruction bundle.

7. (Previously Presented) A method according to claim 3, further comprising compressing the bundle of instructions based at least in part on the number of valid instructions in the bundle.

8. (Currently Amended) A method, comprising:

fetching a bundle of instructions having a complex instruction;

shifting at least one instruction occurring after the complex instruction; and

determining a number of valid instructions by edge detecting the number of valid instructions occurring after the complex instruction and up to a detection of a transition from a valid instruction to an invalid instruction.

9. (Previously Presented) A method according to claim 8, further comprising bundling instructions occurring prior to the complex instruction.

10. (Previously Presented) A method according to claim 8, further comprising executing instructions occurring before the complex instruction.

11. (Previously Presented) A method according to claim 8, further comprising bundling instructions occurring after the complex instruction.

12. (Previously Presented) A method according to claim 8, wherein the step of shifting the instructions comprises compressing the instructions occurring after the complex instruction.

13. (Previously Presented) A method according to claim 8, wherein the step of shifting the instructions comprises compressing the instructions occurring after the complex instruction for a monotonic instruction bundle.

14. (Previously Presented) A method according to claim 8, further comprising executing instructions occurring prior to the complex instruction during a first clock cycle.

15. (Previously Presented) A method according to claim 14, further comprising executing the complex instruction during a second clock cycle.

16. (Previously Presented) A method according to claim 15, wherein the step of shifting the instructions within an instruction bundle occurs while at least one of the instructions occurring prior to the complex instruction are executed and the complex instruction is executed.

17. (Previously Presented) A method, comprising:

fetching an bundle of instructions having a complex instruction;

executing during a first clock cycle valid instructions occurring prior to the complex instruction;

executing the complex instruction during a second clock cycle;

shifting instructions within the bundle occurring after the complex instruction during at least one of the first clock cycle and the second clock cycle;

edge detecting valid instructions occurring after the complex instruction during at least one of the first clock cycle and the second clock cycle; and

executing the valid instructions occurring after the complex instruction during a third clock cycle.

18. (Currently Amended) An apparatus for calculating the number of valid instructions within a microprocessor, comprising:

a mechanism for fetching a bundle of instructions,

an advancing mechanism for advancing instructions along a microprocessor pipeline, and

an edge detection element for detecting a transition from a valid ~~instructions~~ instruction to an invalid instruction within the bundle.

19. (Previously Presented) An apparatus according to claim 18, further comprising a shifting mechanism for shifting at least one instruction based at least in part on the number of valid instructions in the bundle.

20. (Previously Presented) An apparatus according to claim 18, further comprising a compression mechanism, wherein said compression mechanism compressed a bundle of instructions for a monotonic instruction bundle.

21. (Previously Presented) The apparatus according to claim 18, further comprising a compression mechanism, wherein said compression mechanism compressed a bundle of instructions based at least in part on the number of valid instructions in the bundle.

22. (Previously Presented) The apparatus according to claim 18, further comprising a bundling mechanism, wherein said bundling mechanism bundles instructions occurring prior to a complex instruction.

23. (Previously Presented) An apparatus in accordance with claim 18, further comprising an execution mechanism, wherein said execution mechanism executes instructions occurring prior to a complex instruction during a first clock cycle.

24. (Previously presented) An apparatus in accordance with claim 18, further comprising an execution mechanism, wherein said execution mechanism executes a complex instruction during a second clock cycle.